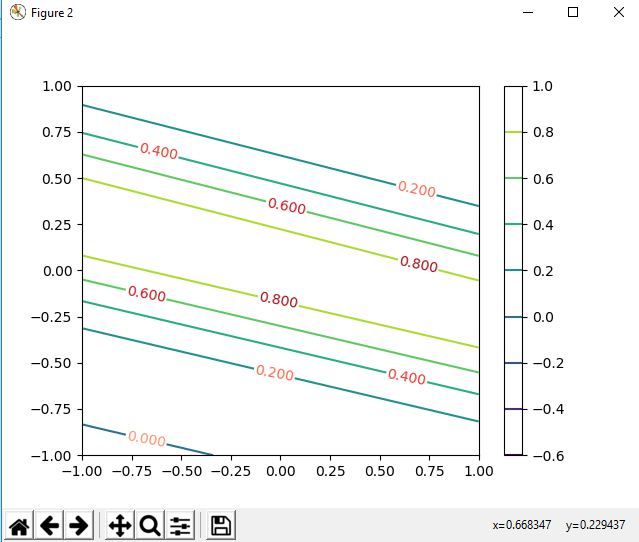
COMP4107 – Neural Networks

Assignment 2 Answers

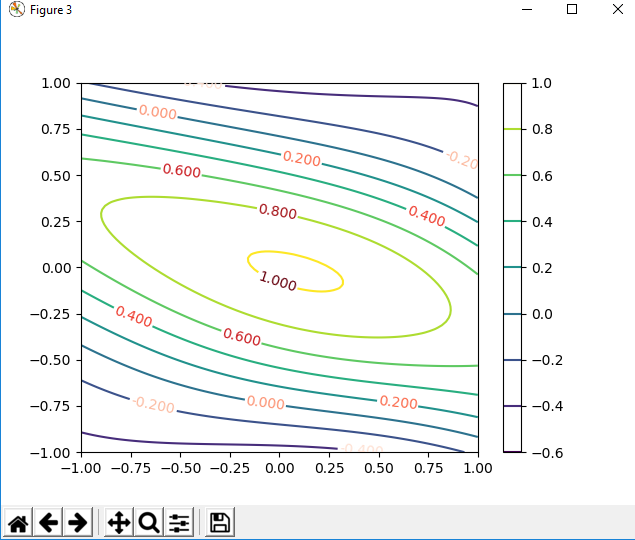
Members: Krystian Wojcicki, Michael Kuang

1. a)

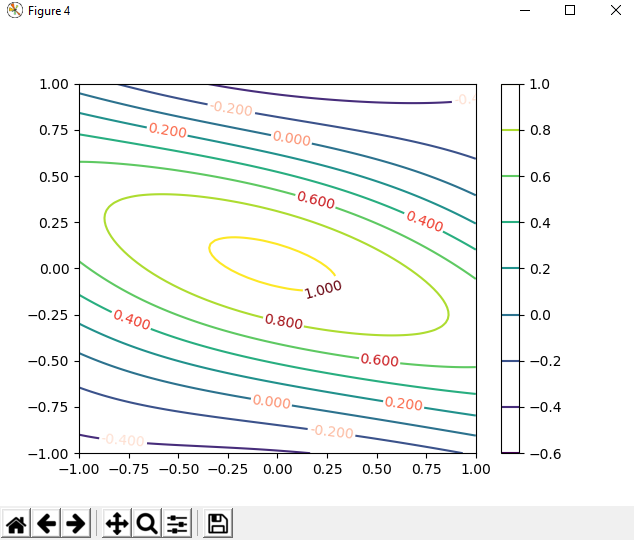
Running `python q1.py a` we get the following graphs. The networks ran for 90 epochs with converge being at MSE of training = 0.02.



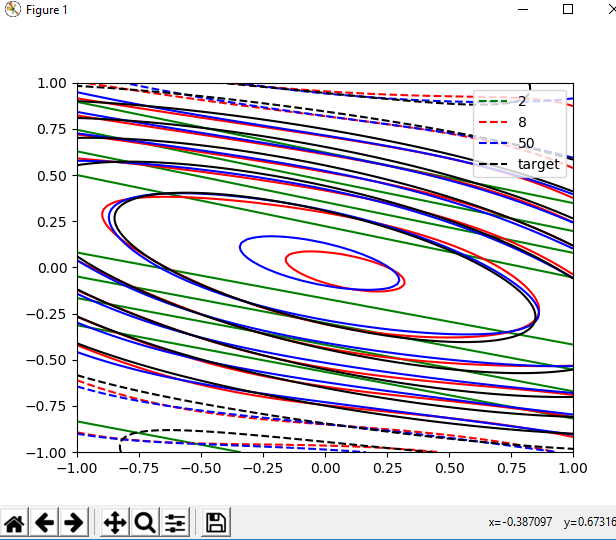
*Contour generated by network with 2 hidden layer neurons*



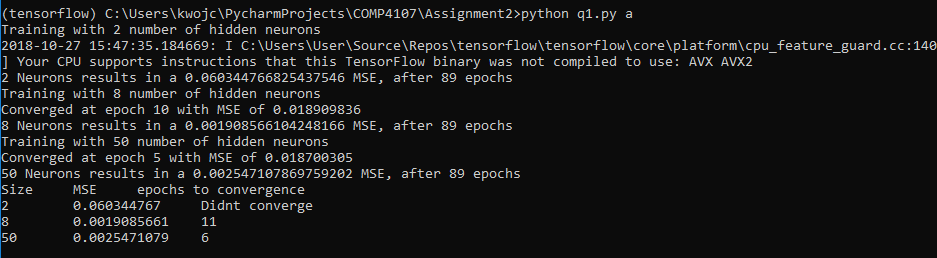
*Contour generated by network with 8 hidden layer neurons*



*Contour generated by network with 8 hidden layer neurons*



*Reproduction of fig 3 from the assignment*

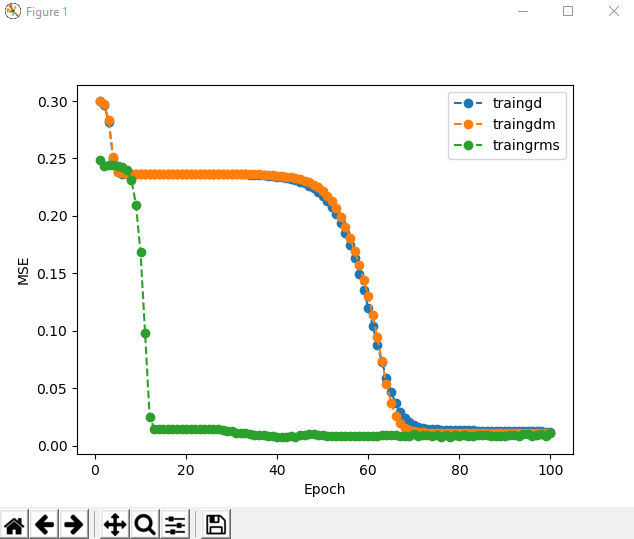


*Table showing hidden layer size and convergence*

The hyperbolic tangent activation function was used. As seen by the reproduction of fig 3, 8 hidden neurons does seem to be the best for modelling the desired function.

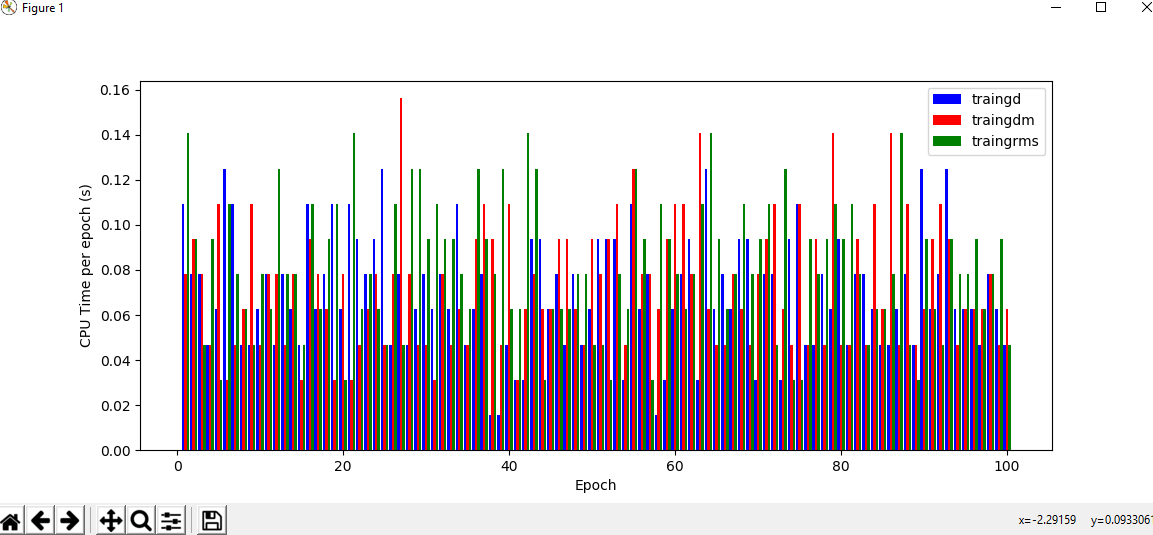
b)

Running `python q1.py b` .

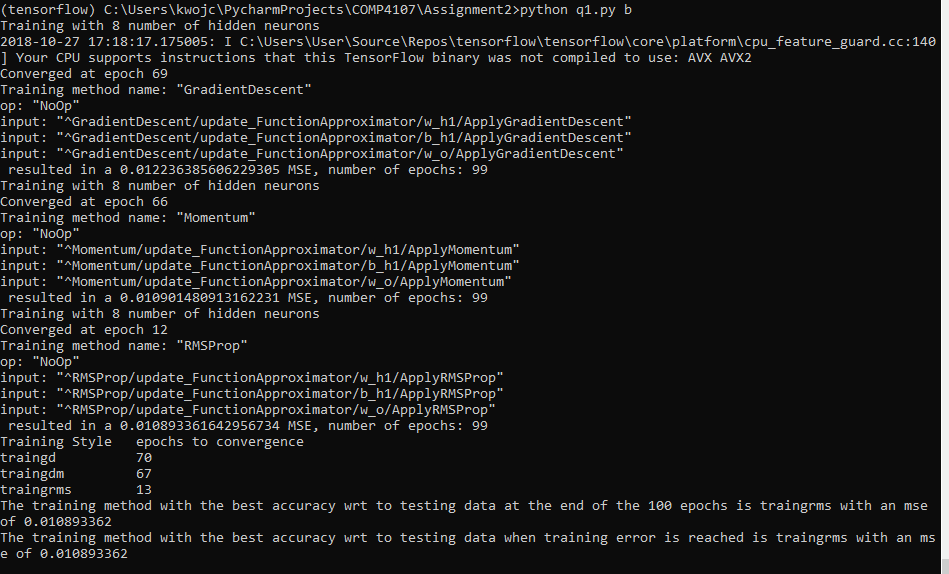


*Graph showing MSE and epoch*

From the above graph we can see that the RMSPropOptimzer did converge the fastest, with MomentumOptimizer slightly quicker than GradientDescentOptimizer. As confirmed by the following table



The CPU time per method fluctuated greatly but it seemed like the more complex the training method the longer it took per epoch.



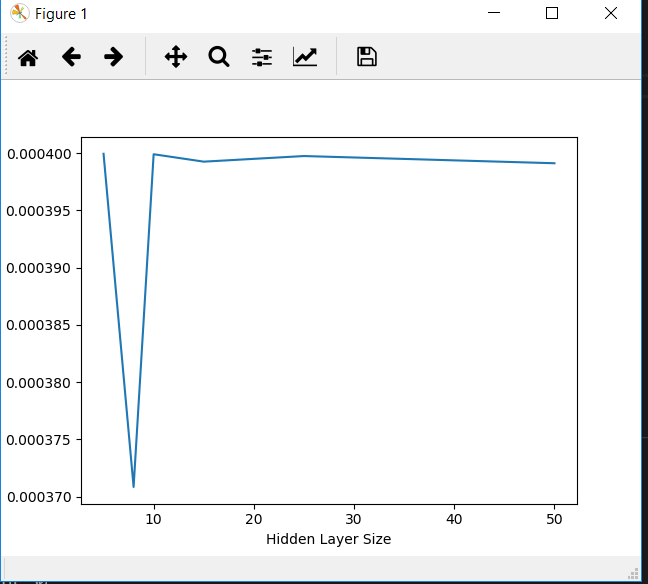
*Table showing convergence with rms the fastest followed by dm, d*

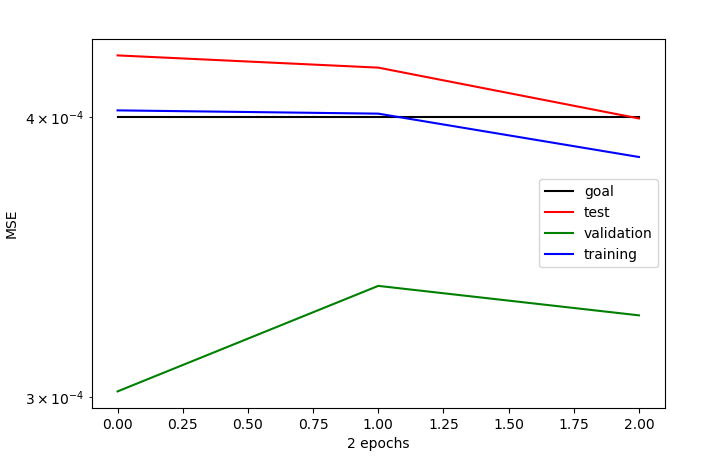
After 100 epochs typically traingrms or traingdm had the best accuracy but sometimes traingd did have the better accuracy. At convergence trainingrms usually and the best accuracy with traingdm sometimes having better accuracy

As expected and explained in the question rms converges the fastest followed by dm and then d.

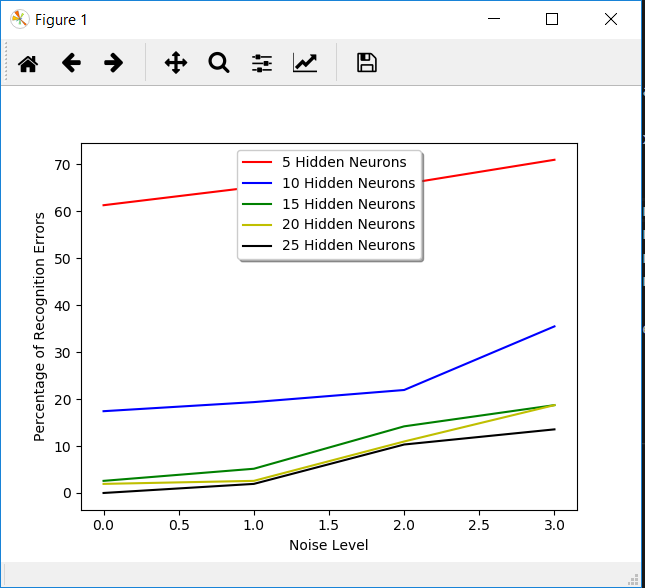
c)

Running `python q1.py c`

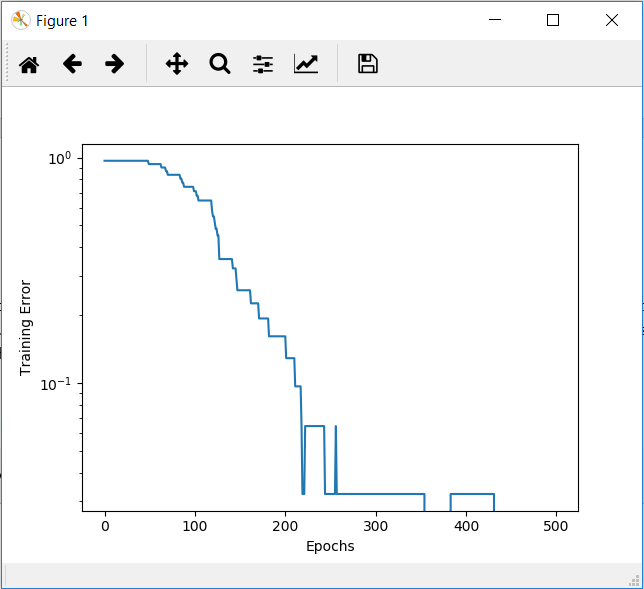




2. a) The first part of q2.py, we ran experiments with hidden neuron numbers in range of 5-25 with increments of 5 for 300 epochs each. The graph below shows the percentage of recognition error for different noise levels and we can see that generally, with each increment of 5 neurons, the percentage of recognition errors decreases. From this we can see that 25 hidden neurons will likely to produce the best accuracy and is the most optimal choice.

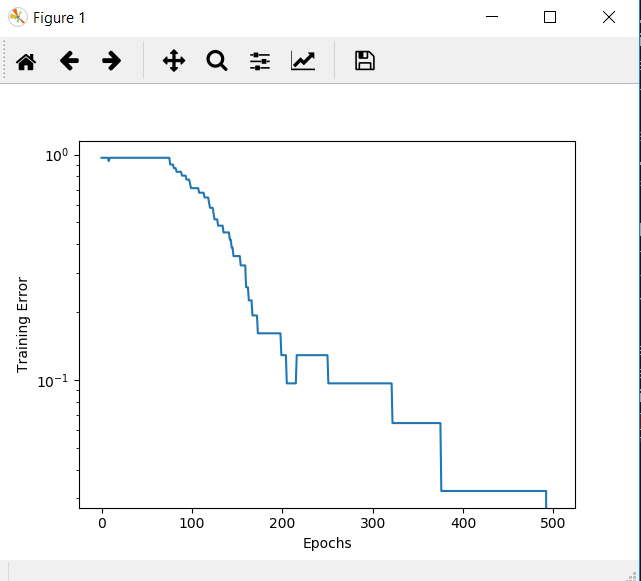


b) In the following graph, we can confirm that Fig.13 from the assignment is a reasonable representation of performance for the optimal number of hidden layer neurons being 25. We can see that at the first hundred epochs, the model does not learn much, then there are a couple sudden drops from approximately 100 to 200 epochs and it flattens out at the end. This makes sense as the network is trained on the ideal data with zero noise. The network should start off at a very high error rate as it should not recognize anything. As more epochs pass, the model should train to the point where it should have very low error rate as shown below.



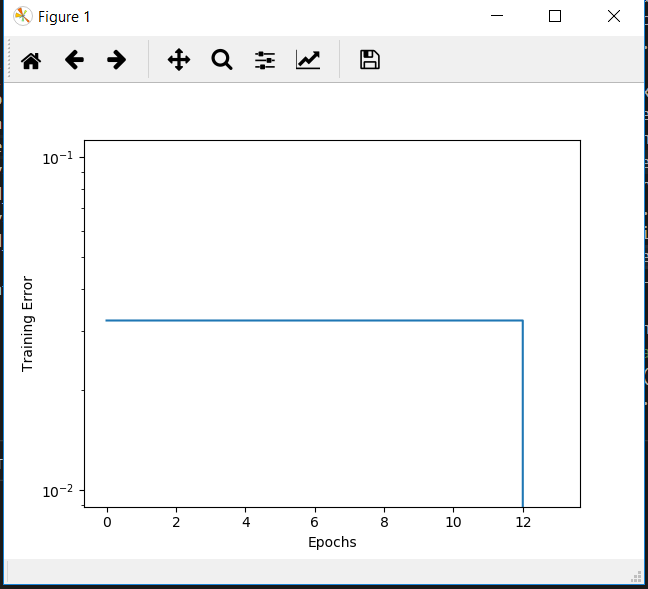
*Graph trained with a hidden layer of 25 hidden neurons*

If we compare this with a network trained on 15 hidden neurons, we can see that with 25, it is able to learn much faster. With approximately 200 epochs, it reaches around 3% error while using a hidden layer of 15 hidden neurons, it takes about 375 epochs.

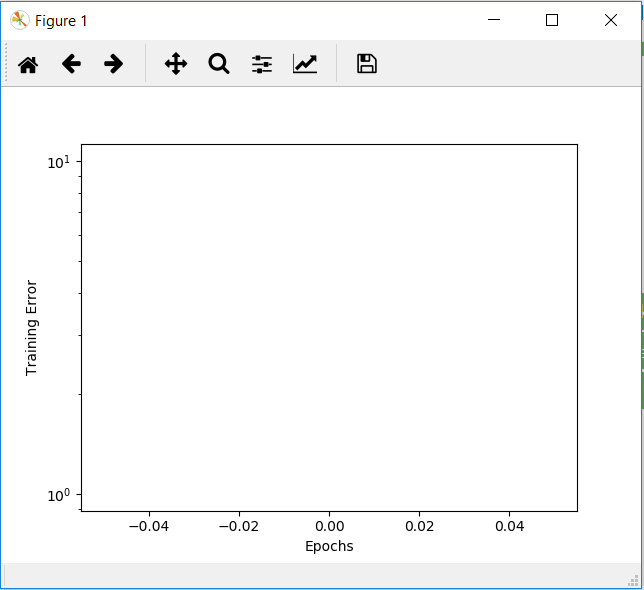


*Graph trained with a hidden layer of 15 hidden neurons*

The figure below is a graph of training error at each epoch after we have trained on the ideal set and noisy data. The graph confirms that after training on the noisy data, it forgot a bit of the noiseless data and as such the error at the beginning is a bit higher, then it drops after training on the noiseless data again.



Please note that there are cases when this graph will show nothing (example below), this is because sometimes the graph is able to retain its understanding of the ideal targets with zero noise even after training on the noisy data.



2 c) As seen in the graph below, we can produce the recognition accuracy shown in Fig. 14. In general, the model trained with noise outperforms the model that was trained without noise.

